Vulnerabilities in Distributed Computer Systems

Vebjørn Moen
The degree philosophiae doctor (PhD)
University of Bergen, Norway
2006

Universitet i Bergen

Contents

Abstract

The society relies more and more on interconnected computers systems and distributed applications. This dissertation considers security in such systems, focusing on how to break the security. Online banking, wireless systems with thin clients, and e-government are covered by the seven papers included in the thesis.

Acknowledgments

I finished my master thesis in 2002, and wanted to continue as a PhD student. Thanks to Professor Tor Helleseth this was possible. Both Professor Lars Knudsen and Professor Helleseth encouraged me to work with computer security. In addition to Professor Helleseth, I have been supervised by Professor Kjell Jørgen Hole. Professor Hole has been my mentor, and made these last years both fun and interesting. I have looked forward to go to work every day, even when I have had to reiterate the fifth or sixth time on one of the articles.

Thanks to all the members of the NoWires Research Group for both work related collaboration and after working hour activities, and a special thanks to my co-authors: Kjell Jørgen Hole, André N. Klingsheim, Håvard Raddum, Kent Inge Fagerland Simonsen, and Thomas Tjøstheim.

I also have to thank Professor Dieter Gollmann for letting me visit his group at Technische Universität Hamburg-Harburg, and thanks to Jens Ove Lauf, Jan Meier, and all the master students for the good discussions and for helping me find my way through the German system.

Last, but not least, thanks to my family, my wife and children for all the love and support.

Practical Security

Practical Security

Vebjørn Moen

1 Introduction

Distributed computer systems play an important role in our daily lives. Online banking, e-commerce, and e-governments are examples of distributed systems that offer more and more services and, thus, grow larger and more complex every day.

We benefit from all these new services, but they can also affect us in negative ways. Privacy concerns with making large databases available on the Internet and increased risk as e-commerce and online banking replace shops and bank branches have to be addressed by building security in. By nature, computer security is a very broad field. It is used to defend a lot of different data on many different platforms. Human interaction adds to the complexity and for most applications both national and international laws and regulations play an important role.

We can make some general observations about security: Since it is easier to attack a system than to defend it, the attacker has an advantage. To properly defend an asset, one has to defend it against all possible attacks. However, to break a system it is enough to find one exploitable vulnerability, and it does not help if a system is unbreakable at one level, if it is insecure at another level. Often one vulnerability also makes it easier to break the rest of the security of a product. These observations lead to the axiom that security is not stronger than the weakest link.

To make sure that we do not create systems that provide a false sense of security, security experts invest a lot of effort into finding vulnerabilities [1, 2], which is also the main focus of the papers in this thesis.

The next sections are organized as follows: Section 2 gives an introduction to what security is and how to understand the security of a system, Section 3 discusses how security should be managed—including a discussion of what creates vulnerabilities, Section 4 gives an overview of the papers included in this dissertation, and Section 5 concludes the first part of the thesis. The second and main part of the dissertation consists of a collection of papers.

2 What is security?

The main information security elements are *confidentiality*, *integrity* and *availability* [3, p. 5]. Confidentiality ensures that an attacker cannot read the protected information. In a system with integrity, any unauthorized modifications of the content will be detected. The availability goal is to keep services up and running and to make sure that all users can access the services.

Computer security can be defined as the protection of computer systems against actions which violate the desired security elements. To be able to provide computer security, we would like to specify and implement a security policy, defining actions as legal or illegal. Protection against attacks is then reduced to access control, of actions such as read, write, and delete. The access control requires an entity authentication or identification, i.e. corroboration of the identity of a person, a computer terminal, or a credit card [4]. Since some actions can be considered hostile in one computer system, and legal and normal in another, we need to know the details of the application/system before we create the security policy. Therefore, one definition or policy cannot fit every application, and we have no absolute definition of what security is.

Cryptographic algorithms lay the foundation for most of the technologies used to solve computer security problems. A good understanding of cryptology is needed to correctly design and implement security technologies. From the end of the 1970s and up till approximately year 2000 the common view reflected in many books such as [5] and [6], was that *when* we got the cryptography right every computer security problem would be solved.

2.1 Security problems

Attacks on computer systems are exploits of an imperfect specification or implementation of a security policy. We call security vulnerabilities in the design for *flaws*, and vulnerabilities in the implementation for *bugs* [7].

Practical experience during the last few years has showed that the problems with computer security are much broader than the underlying algorithms. Examples include: buffer overflow, Structured Query Language (SQL) injection, phishing, side channel attacks, virus, trojans, and so on. All these security problems that cost the society large amounts of money each year [8], cannot be solved by cryptography alone. The insecurities are caused by flaws within the design and bugs in the implementation, and broken abstractions when ideas and designs travel between different realms. Software applications are made to simplify laborious tasks, and solutions have to solve problems in different fields. The design and implementation have to consider the economic bottom line, the legal aspects, and the viewpoint of the end user. Developing an application that can perform the intended task can be difficult enough, but it gets even more difficult when the whole system has to be able to handle any kind of malicious input.

The engineering of building bridges has a lot in common with computer security. A secure bridge has to be able to handle the wear and tear of normal use, and external influence such as wind and rain. If the bridge is badly engineered it can lead to a bridge failure, e.g. caused by resonance [9]. Therefore, the engineering principles are similar—such as investigate, design and implement good defenses against the threats. However, the threats against a computer system are of a different nature. The main difference can be found in the characteristic of the assets, a bridge is a physical object and is only vulnerable to physical attacks, but a computer system consists of both physical and nonphysical assets. Nonphysical assets, such as information, up-time, or the number representing the balance on an account, are vulnerable to a whole new range of attacks. Systems connected to the Internet are especially vulnerable as they can be attacked from any computer connected to the Internet. Very few people try to

bring down a bridge for fun, but online resources have to sustain daily attacks from malicious hackers who are trying to break the security for fun or profit.

2.2 Improving security

In cryptology, attacks are considered to be a reasonable way of evolving the security. Full-scale computer security solutions are easier to attack than cryptographic primitives, since they are bigger, more complex, and can be attacked on any of the many layers of abstractions.

The common perception in cryptology is that anyone who are designing their own cryptosystem, should consider all known attacks against such a system, but also try to find new attacks. We argue that the methodology is suitable also for computer security systems.

We study real systems, and the evaluation of such systems often starts with finding vulnerabilities. If you only come up with theoretical results, it is usually difficult to convince the owner of the system that there is a problem, but one simple attack quickly and clearly illustrates the problem in the design or implementation. Looking for attacks is a good starting point both when designing a new system and analyzing an existing system.

From Sun-Tzu's *Art of War*: *"If you know the enemy and know yourself, you need not fear the result of a hundred battles. If you know yourself but not the enemy, for every victory gained you will also suffer a defeat. If you know neither the enemy nor yourself, you will succumb in every battle."* When building a secure computer system Sun-Tzu's wisdom tells us that without knowing our enemy and the attacks our enemy can launch against us, the attacker will succeed with his attacks on our systems. We need to know and understand what we are defending against. We need to understand the mindset and the toolbox of the attacker, and when we analyze security systems we need to think as the attacker. When we have the knowledge of our attackers, we are more capable to decide which defenses we have to spend more resources on.

2.3 Computer security—academic discipline or business secret?

Many companies consider secrecy about their system to be an important part of the security. We have seen how the banking industry in Norway keep even their most basic descriptions of security systems confidential, and disclosures of vulnerabilities are regarded as very harmful. Not because of the vulnerabilities, but because of bad press and loss of reputation. *Security by obscurity* can be regarded in different ways. Either the systems are more secure because the owners keep everything secret, or the systems are more insecure. A system where no details are public, can have bugs that nobody knows about. It can therefore run for several years without these bugs being a problem, and therefore the company can save money on not fixing the bugs and flaws. It can also be in production several years with a bug that only a few people know about and are able to misuse. Basing security on secrecy is a flaw in the design, since information will leak about the system over the years. Disgruntled employees can also become malicious attackers.

In addition, there are also other problems when companies refuse to share information about their systems. It creates a closed developer community where

there will be fewer new ideas and solutions, since there can be few ideas traded to and from a closed group. This creates a mode of thought known as 'group-think'. Quality assurance will also be less efficient in a closed group, especially if testing and evaluation are done by the same people who created the system. The solution is probably not to tell the world the details of security routines and implementation details, but having an open discussion about development methods and design with external experts to improve security.

Industry's security by obscurity prevents the academic world from learning about good research problems. Not only does the policy harm research, but also the education of students. When students learn nothing about the common industry problems at the university, the industry is left to educate their own employees. As a result, they tend to maintain a limited and static view of vulnerabilities and security techniques.

We have also seen how problems arise when companies that keep their system secret are allowed to stand in court and claim that their systems are secure. When the court believes such a claim, it is easy to argue that any incident must be caused by wrong behavior by the users.

3 Measure and manage security

In 1883, Lord Kelvin stated "*I often say that when you can measure what you are speaking about, and express it in numbers, you know something about it; but when you cannot measure it, when you cannot express it in numbers, your knowledge is of a meager and unsatisfactory kind; it may be the beginning of knowledge, but you have scarcely in your thoughts advanced to the state of Science, whatever the matter may be.*" [10]

In cryptology there are some cryptosystems with a proof of security, e.g. Rabin public-key encryption [4, p. 292], but as Lars Knudsen has said "*If it is provably secure, it is probably not.*" The Rabin public-key cryptosystem is provably secure against one type of attack, but is easily broken if an attacker is allowed to choose ciphertexts and get them decrypted. Proving that cryptographic primitives are secure against every attack is difficult, and provable secure cryptosystems get broken by nullifying the assumptions. When we consider an application, then proving anything about the security is very difficult, and perhaps impossible. Too many assumptions need to be made, and these assumptions are the attack points.

Provable security is probably impossible, but how about rating the security of a product on a scale from 1 to 10? The challenge is to be able to say more about the security of an application other than that the security is broken (proved by a practical attack), the security is untested, or no attacks are found after an analysis. Any measured security of an application will be meaningless if an exploitable flaw or bug is found.

The problem of making provable secure systems are linked to the lack of a good definition of security. For one application we need to decide what we want to be secure against, and defenses against one attack might make the application vulnerable to other attacks. One example is the protection against brute-force attacks on online banks described in [11], where the attacked customer accounts are closed down after a small number of unsuccessful login attempts. A protection against brute-force attacks ends up as a vulnerability to a Denial-of-Service

(DoS) attack.

Since we cannot measure or prove what a secure application is, we try to say what it is not. Karl Popper discussed the concept of falsification [12], where an expression such as "All swans are white" cannot be proved, because it implies observing every swan in the present, past, and future. However, it is easy to falsify; that only requires finding one swan that is e.g. black. The same idea can be applied to security. Claiming that a program is secure requires analyzing it for every possible security bug and flaw, both known and unknown. However, we only need to find one vulnerability to say that a program or system is not secure. Following the argument, a security evaluation of a product cannot look for reasons to conclude that the product is secure. A security analysis must consider possible attacks, both old and new, and how to protect against these attacks.

3.1 What creates vulnerabilities?

According to [7] there is a trinity of trouble in computer software security: complexity, connectivity, and extensibility. The more complex a system is, the more difficult it is to analyze it, and therefore it is more likely that bugs and flaws exist but are missed in a security analysis.

Distributed and connected software are challenging to design and implement properly. The system designer must consider completely new attacks that were not a problem when developing applications for stand-alone computers. The connectivity therefore adds extra complexity to the application.

To incrementally evolve the system functionality, modern applications are often extensible with plug-ins, scripting, and updates. System designers and users like extensible systems since they are dynamic and flexible and can be adapted to do new tasks. Attackers like extensible systems for the same reasons. Extensibility is closely related to connectivity, and the extensibility feature certainly adds to the complexity of the software.

In [13], we show that many e-government projects are vulnerable to simple, well-known Web application attacks, and that industrialized countries have more vulnerable Web pages than under-developed countries. We further show that industrialized countries have more pages created with more types of technologies. The results support the idea of the trinity of trouble in computer software security.

If we buy the argument that complexity, connectivity, and extensibility are the cause of vulnerabilities, what can we do about it? Create less complex, not connected software with no possibility to extend the application? In some cases simpler solutions make sense, but for most applications these features are exactly what are needed to perform the intended tasks.

3.2 Who is the attacker?

As previously mentioned, there exist cryptosystems that are provable secure. Despite the proof of security there exist practical attacks against some of these systems. The question in a threat analysis is not whether the system is secure or not, it is what and who the system is secure against.

Consider a banking system. It is quite clear what such a system has to protect, it is the account holders' assets. In early banking systems it was also

clear who the customers had to trust, namely the local bank that held and protected their money. There were two kinds of attackers, the bank robbers and the rogue employee, and they were bound by physical limitations. To steal money, the perpetrator had to be in the same location as the money, and if money was missing from the bank vault, and there were no signs of a break-in, then an employee had to be responsible.

There are still two kinds of attackers in modern banking, but now the bank robbers or rogue employees do not have to be in the same place as the money. A break-in to a computer system does not necessarily leave any sign, and it can be facilitated by vulnerabilities in the software.

To defend against insider attacks encryption and authentication keys are created and kept in crypto-modules, and strong cryptography is used to protect transactions. However, [14] shows that the banking industry has a history of phantom withdrawals, break-ins, and documented [15] vulnerabilities in the APIs of the crypto-modules. Thomas Whiteside [16] writes about a supposedly real attack where an embezzler transferred small amounts from many accounts into his own account. There is also a legend of an automated variation of the same attack, where the attacker collected the roundoff of pennies in calculation of e.g. interests, and transferred the money into an account under his control. The history of documented vulnerabilities shows that banks and any owner of a distributed computer system need to have a good understanding of which risks that are worth taking, and which risks that need to be mitigated.

3.3 Mitigating risk

The management of a company is responsible for the bottom line, maximizing income and minimizing expenses. Security is for most companies only a necessary expense, and management will therefore influence how discovered vulnerabilities and risks will be handled. It is bad business to fix a vulnerability if it is cheaper to take the risk of leaving it unfixed, and there are also other methods of mitigating risk than to fix the vulnerabilities. Transferring the risk to different entities through outsourcing, contracts, and terms of use can be effective business "solutions" to difficult computer security problems.

We have seen how online banking systems, even though they save the banks a lot of money on handling accounts, have transferred more risk to the customer. Ross Anderson [17] describes why letting the banks transfer risk to the customer is a bad idea, and in [14] he claims that making the banks responsible for the risk associated new technologies helps promoting security technologies. Risk is an inherent part of every system, and especially distributed software applications. Changes to the system, and introduction of new technology changes the risk. It can also change who have to take the risk. Anderson *et al.* [18] have also shown how the introduction of chip and pin to replace a system of magnetic stripe cards and signatures can be used to transfer more of the risk that banks traditionally have carried over to the customers.

4 Summary of the Thesis

All the papers in this thesis consider security in client-server systems—from cryptologic attacks to design processes and how to develop secure software.

Three of the papers discuss banking systems, three papers explore wireless systems with resource-constrained devices, and one paper consider e-government systems. Five of the papers [11, 19, 20, 21, 13] have been published in refereed journals and conferences, while the last two papers [22, 23] have been submitted for publication.

The main focus of the thesis is analysis of security in distributed computer systems, and mainly on how to attack such systems. However, the papers also suggests how one could fix such vulnerabilities.

4.1 Vulnerabilities in online banks [11]

Online banks have become very popular. The banks save money by letting the customers manage their own transactions, and the customer has the benefit of easy access to updated information about their accounts.

The Norwegian banks tend to claim that Internet banking is absolutely secure if used correctly by the customers. We show that login schemes based on identifiers with a structure and Personal Identification Numbers (PINs) are insecure against a simple brute-force attack. The idea behind the brute-force attack is not to guess the PIN of one customer, but to guess the PIN belonging to a random customer. One example of the attack is to try a few PINs for each customer in a given bank. To have a high probability of success, the attack requires that the bank has more customers than possible PIN values, and that the identifiers are possible to generate. Social Security Numbers (SSNs) and bank account numbers are good examples of such generable values. The attack works even if the online banks use one-time PIN values from lists or PIN calculators if the length of the PINs is small enough.

4.2 Case study: online banking security [19]

We wanted to find out if other solutions were affected by the brute-force attack described in [11]. Several banks use PIN calculators to calculate one-time PINs. In general, PIN calculators are used to create a two-factor authentication scheme, something you have and something you know. A two-factor scheme should be more secure than the static 4 digit PIN solution we analyzed in [11]. However, we found out that because of difficulty in clock synchronizing between the calculator and the server, the server accepts more than one PIN. One of the banks accepted any PIN in a window of 19 consecutive PINs, which drastically increases the efficiency of the brute-force attack compared to a system with a static PIN of same length.

By entering a wrong PIN for an account more than, say, 3 or 4 times, the account will be locked. An attacker can thus combine DoS and brute-force attacks to maximize damage, and increase the likelihood of getting away with the money in the confusion.

Despite claims from Norwegian banks that online banking is completely secure, we found simple and practical brute-force and DoS attacks on several banks. It is likely that the systems had the weaknesses because the banks based their solution on the experience they gained from ATM systems, where these attacks are not practical. We believe that these problems could have been avoided with a proper design phase including evaluation by independent experts.

When trying to inform the banks about the weaknesses, we discovered that they are not interested in improving the security, only in maintaining an impression of secure products. We also learned that the employees of banks are not allowed to discuss security solutions with external experts.

4.3 Lessons from the Norwegian ATM system [22]

Every year around 400 complaints about misused ATM cards are considered by an institution in Norway called "Bankklagenemda". Many cases are dismissed with the argument that the ATM system is secure and that the victim must have kept the PIN together with the card, or in some other way disclosed it. According to "Bankklagenemda" there are no other explanations for how the PIN was entered correctly by an attacker.

One of these cases went on to court, where the representatives from the bank claimed that their system is secure and that disclosing details about it will make it less secure. The most important claim was that it is impossible to brute-force the PIN, based on the information available on the card, in the period of time from the card was stolen to the card was misused in an ATM. We describe an attack scenario where finding the PIN is done in a few seconds, given that the attackers have carried out pre-calculations to find the bank specific cryptographic key used to calculate the PIN verification value found on the magnetic strip of the card. Today the key used to calculate the PIN verification value is a 3-DES 112-bit key, but until 2000 many banks used 56-bit single-DES keys. The 112-bit keys are too strong, but the 56-bit keys are possible to find using sufficient computational power in a brute-force attack.

The described security issues associated with the ATM system, led us to question how to create stronger systems. We suggest that new and open development processes will improve the security in banking systems, and make it more clear who is responsible for which risk factors.

4.4 Weaknesses in the Temporal Key Hash of WPA [20]

The security scheme Wired Equivalent Privacy (WEP) is a part of the IEEE 802.11 standard for wireless communication. WEP has been shown to be weak [24], and practical attacks exist on all its security features.

To address the weaknesses of WEP, Wi-Fi Protected Access (WPA) was put forward to secure the communication in 802.11 networks. We show that the WPA protocol also has weaknesses, one of them being that it is possible to find the 128-bit master key in WPA with the work equivalent to 2^{105} RC4 encryptions, reducing the effective key size from 128 to 105 bits.

The master key is used to derive per-packet RC4 keys. Assuming we can find some sequential packet keys, we can recover the master key by attacking the key generation algorithm. In essence the attack is an exhaustive search, but we can split the search into separate searches for parts of the master key. Given four RC4 keys the master key can be recovered in 6-7 minutes on an Intel Pentium 4 2.53 GHz with a workload equivalent to $\mathcal{O}\left(2^{31}\right)$ temporal key hash operations. Given only two RC4 keys the workload is $\mathcal{O}\left(2^{38}\right)$ and master key recovery takes approximately 15 hours.

4.5 Attack on Sun's MIDP reference implementation of SSL [21]

It seems that every time applications are developed for new platforms, old security problems appear again. This time it is the generation of random data used to create symmetric keys. Just as Goldberg and Wagner [25] showed that the early version of SSL in Netscape was seeded with time, Sun's Mobile Information Device Profile (MIDP) reference implementation of SSL use time as seed to create keys, which drastically reduce the effective key size.

We show how to implement a practical attack against Sun's SSL implementation, which can find the SSL premaster secret based on the transmitted handshake messages. The premaster secret generated by the emulator in Sun Java 2, Micro Edition (J2ME) Wireless Toolkit 2.1 was recovered within one second. However, it is unclear if the attack is also a problem in implementations of MIDP in mobile phones, because we did not recover an SSL key from any of the mobile phones we tried the attack on. We can only speculate how mobile phone manufacturers creates pseudo-random numbers on the devices, since very little information is available.

4.6 Secure networked J2ME applications [23]

The article [23] is based on the experiences we made during the design phase of a commercial application [26] for smart-phones. The application should be able to receive, store and show medical information. In addition, it should run on as many smart-phones as possible. Due to legal considerations, the medical information had to be encrypted during transport, preferably stored encrypted on the phone, and it could only be stored on the server for a short time. Based on market information, we chose J2ME as the development platform. We did not have the desire or resources to implement our own encryption routines, so we targeted MIDP 2.0 devices since they offer HTTPS.

During the project we tested several smart-phones from Nokia, Motorola, Samsung, and SonyEricsson. We were surprised to see that several of the phones had serious bugs affecting how HTTPS connections and certificates were handled, making it impossible to support all MIDP 2.0 phones with MIDP 2.0 compliant applications.

Implementations of MIDP 2.0 contain too many bugs, and do not offer enough security functionality to create applications with the same security as applications for Java 2, Standard Edition. Therefore, we also discuss the forthcoming Security And Trust Services API (SATSA), which will be shipped with future mobile phones, that offers encryption and signing functions that can be used to create even more secure applications for smart-phones.

4.7 Vulnerabilities in e-governments [13]

To get an impression of the maturity of e-governments we checked the Web pages of governments around the world for simple Web application attacks. The attacks we looked at were SQL injection and Cross Site Scripting (XSS). We chose these attacks since they are well known and understood, and relatively easy to defend against.

In [13] we show that more than 80% of the e-governments in the world are vulnerable to these attacks, and that industrialized countries are more vulnerable than under-developed countries.

In addition, the paper also describes some malicious data mining possibilities in the Norwegian e-government. Several Web applications allow an attacker to filter SSNs. Some pages also make it possible to find the name of a person with known or guessed SSN. The information gathered on the e-government pages can be used for identity theft.

5 Conclusion

In this thesis we have explored security in real life systems, and we have discovered how challenging it is for independent researchers in the security field to present results in a constructive and fruitful manner. Disclosed vulnerabilities increase the risk of attacks, and can potentially cost companies a lot of money. Still, if no independent research is done on the security of real life systems the knowledge of the systems' security is completely controlled by companies. Since our work is funded by tax payers, we feel an obligation to evaluate national systems.

Based on how vulnerabilities have been managed in Norwegian banking systems, we believe that independent research can balance the knowledge between customers and banks, and help customers to take more educated decisions and avoid becoming victims of the banks' security-by-obscurity policy.

In [22], we looked at a court case were a Norwegian bank was sued by a customer because of stolen and misused credit cards. In the court, the bank claimed that it could not disclose any details about their system, since that would make the system more insecure. However, they were prepared to put some of their top people on the stand to testify that their system is secure. We claim that this security-by-obscurity attitude does not make bank systems more secure, and that it is harmful to the customers. It is not unlikely that similar cases to the one described in [22] will appear for online bank systems in the future.

Perfect security does not exist, and it is important to consider the risks we take when we develop large and complex systems for new platforms, e.g. wireless thin clients for banking systems. Without a proper infrastructure for identification and authentication, supporting all platforms, online banking and e-government services will continue to be vulnerable to simple attacks against the authentication schemes. The solutions to the computer security problems are not only technical, since security solutions are also influenced by laws, regulations, psychology and human-computer interaction. Perhaps we will witness a paradigm shift in computer security, where researchers have to focus on economy and law in addition to computers and algorithms.

References

[1] E. Rescorla, "Is finding security holes a good idea?" *IEEE Security & Privacy*, vol. 3, no. 1, pp. 14–19, 2005.

[2] D. Farmer and W. Z. Venema, "Improving the security of your site by breaking into it," Posted to Usenet, 1993.

[3] D. Gollmann, *Computer Security.* John Wiley and Sons Ltd, 1998.

[4] A. Menezes, P. van Oorschot, and S. Vanstone, *Handbook of Applied Cryptography.* CRC Press, 1996.

[5] S. Singh, *The Code Book: The Science of Secrecy from Ancient Egypt to Quantum Cryptography.* Doubleday, 1999.

[6] B. Schneier, *Applied Cryptography, Second Edition.* John Wiley & Sons, 1996.

[7] G. Hoglund and G. McGraw, *Exploiting Software: How to Break Code.* Addison-Wesley, 2004.

[8] L. A. Gordon, M. P. Loeb, W. Lucyshyn, and R. Richardson, "2005 CSI/FBI computer crime and security survey," 2005. [Online]. Available: http://www.gocsi.com/

[9] K. Billah and R. Scanlan, "Resonance, Tacoma Narrows bridge failure, and undergraduate physics textbooks," *American Journal of Physics*, vol. 59, no. 2, pp. 118–124, 1991.

[10] W. Thomson, *Popular Lectures and Addresses Vol. 1.* MacMilllan, 1891.

[11] T. Tjøstheim and V. Moen, "Vulnerabilities in online banks," in *10th Nordic Workshop on Secure IT-systems (Nordsec 2005)*, 2005.

[12] K. Popper, *The Logic of Scientific Discovery.* Basic Books, 1959.

[13] V. Moen, A. N. Klingsheim, K. I. F. Simonsen, and K. J. Hole, "Vulnerabilities in e-governments," in *2nd International Conference on Global E-Security (ICGeS-06)*, 2006.

[14] R. Anderson, "Why cryptosystems fail, from communications of the ACM, november, 1994," in *William Stallings, Practical Cryptography for Data Internetworks.* IEEE Computer Society Press, 1996. [Online]. Available: citeseer.ist.psu.edu/anderson94why.html

[15] M. Bond and R. Anderson, "API-level attacks on embedded systems," *Computer*, vol. 34, no. 10, pp. 67–75, 2001. [Online]. Available: citeseer.ist.psu.edu/bond01apilevel.html

[16] T. Whiteside, *Computer Capers: Tales of Electronic Thievery, Embezzlement, and Fraud.* Ty Crowell Co, 1978.

[17] R. Anderson, "Why information security is hard - an economic perspective," 2001. [Online]. Available: citeseer.ist.psu.edu/anderson01why.html

[18] R. Anderson, M. Bond, and S. J. Murdoch, "Chip and spin," 2006. [Online]. Available: www.chipandspin.co.uk/

[19] K. J. Hole, V. Moen, and T. Tjøstheim, "Case study: Online banking security." *IEEE Security & Privacy*, vol. 4, no. 2, pp. 14–20, 2006.

[20] V. Moen, H. Raddum, and K. J. Hole, "Weaknesses in the temporal key hash of WPA," *ACM SIGMOBILE Mobile Computing and Communications Review*, vol. 8, no. 2, pp. 76–83, 2004.

[21] K. I. F. Simonsen, V. Moen, and K. J. Hole, "Attack on sun's MIDP reference implementation of SSL," in *10th Nordic Workshop on Secure IT-systems (Nordsec 2005)*, 2005.

[22] K. J. Hole, V. Moen, and A. N. Klingsheim, "Lessons from the Norwegian ATM system," 2006, submitted to IEEE Security & Privacy.

[23] A. N. Klingsheim, V. Moen, and K. J. Hole, "Secure networked J2ME applications," 2006, submitted to IEEE Computer.

[24] N. Borisov, I. Goldberg, and D. Wagner, "Intercepting mobile communications: The insecurity of 802.11," in *MOBICOM 2001*, 2001.

[25] I. Goldberg and D. Wagner, "Randomness and the Netscape browser," *Dr. Dobb's Journal*, pp. 66–70, January 1996.

[26] World Medical Center, last visited: June 13th, 2006. [Online]. Available: http://www.world-medical-center.com/